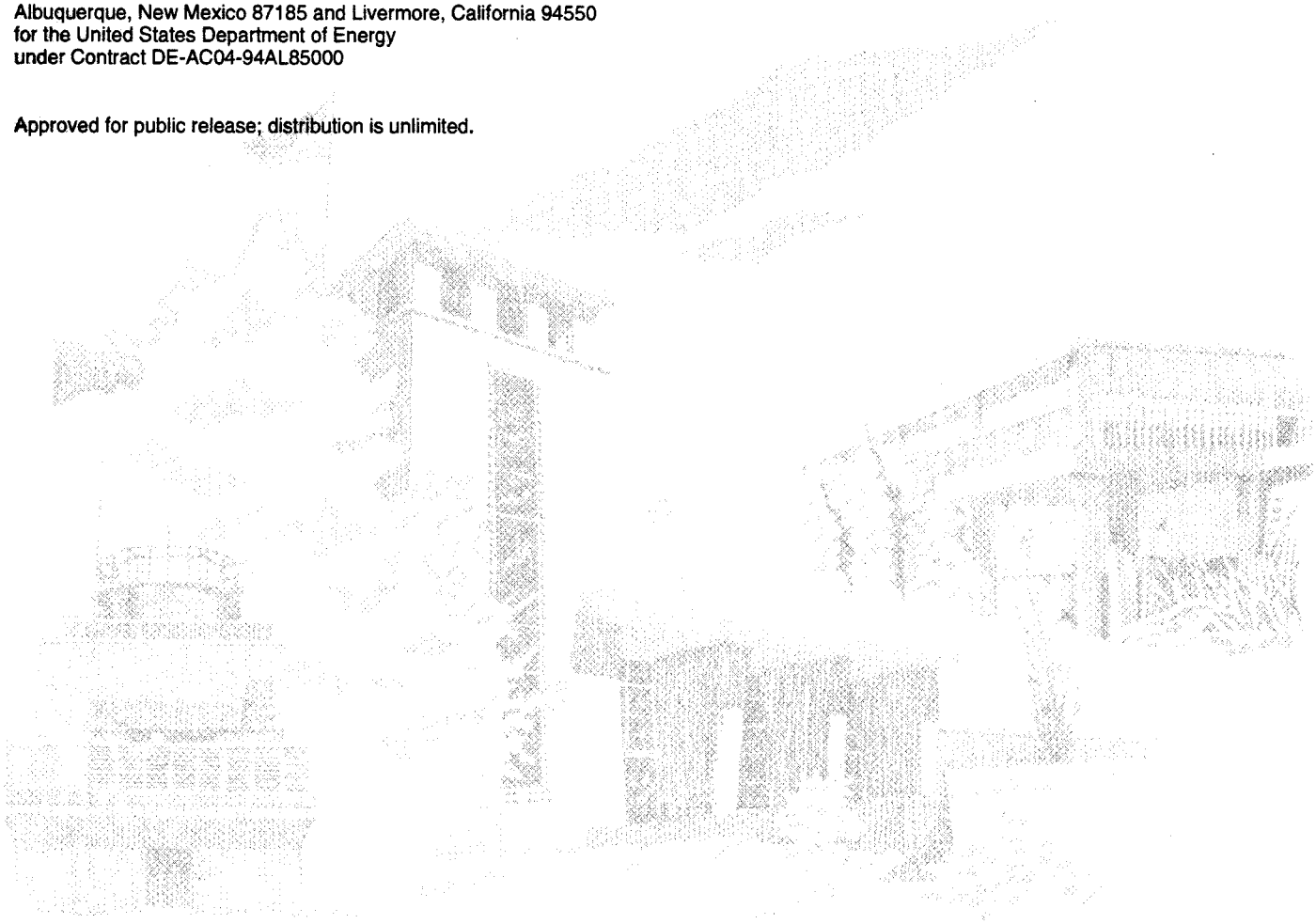
SANDIA REPORT

SAND96–3003
Unlimited Release
Printed December 1996

Data Zooming -- A New Physics for Information Navigation

J. F. Mareda, G. S. Davidson

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-94AL85000



SF2900Q(8-81)

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
Office of Scientific and Technical Information
PO Box 62
Oak Ridge, TN 37831

Prices available from (615) 576-8401, FTS 626-8401

Available to the public from
National Technical Information Service
US Department of Commerce
5285 Port Royal Rd
Springfield, VA 22161

NTIS price codes
Printed copy: A03
Microfiche copy: A01

SAND96-3003
Unlimited Release
Printed December 1996

DATA ZOOMING -- A NEW PHYSICS FOR INFORMATION NAVIGATION

J. F. Mareda
Computer Architectures Department

G. S. Davidson
Computer Architectures Department

Sandia National Laboratories
Albuquerque, NM 87185-0318

Abstract

This research project, with Ben Bederson and Jim Holland at UNM, used Pad++, a user interface originally developed by Jim Holland. The objective was to explore the utility of that user interface to large databases of information such as those found on the World Wide Web. A web browser based on Pad++ was developed in the first year of this project. The first year results, including the human factors were documented in a video and were presented at a SNL-wide seminar. The second year of this research project focused on applying the results of the first year research. The work in the second year involve using Pad++ as a basis for tools to manage large complicated web sites. Pad++ is ideally suited to this complex activity. A prototype was developed, which presents Web relationships in 3D hyperspace, following research from the Geometry Center at the University of Minnesota. Various human factors studies were completed, which indicate Pad++ web browsers allow users to comprehend 23% faster than when using Netscape.

Data Zooming -- A New Physics for Information Navigation

Abstract

This research project, with Ben Bederson and Jim Holland at UNM, used Pad++, a user interface originally developed by Jim Holland. The objective was to explore the utility of that user interface to large databases of information such as those found on the World Wide Web. A web browser based on Pad++ was developed in the first year of this project. The first year results, including the human factors studies were documented in a video and were presented at a SNL-wide seminar. The second year of this research project focused on applying the results of the first year research. The work in the second year involve using Pad++ as a basis for tools to manage large complicated web sites. Pad++ is ideally suited to this complex activity. A prototype was developed, which presents Web relationships in 3D hyperspace, following research from the Geometry Center at the University of Minnesota. Various human factors studies were completed, which indicate Pad++ web browsers allow users to comprehend 23% faster than when using Netscape.

Introduction

This document reports a two year LDRD study on a new user interface, Pad++. This work was undertaken jointly with Ben Bederson and James Holland at UNM, using the Pad++ tool originally developed by Holland. The research extended previous concepts to create a unique Web browser, which was carefully analyzed by the Sandia human factors group. The unique Pad++ browser was found to be superior to Netscape for various experimental tasks.

This report discusses the progress in the first year, then presents the progress in the final year of the project. The human factors studies and results are briefly summarized before concluding. However, Appendix I contains the details of the human factors study, results, and conclusions. Appendix II contains a list of the project members and their responsibilities. Appendix III documents the procedure for installing Pad++.

Project Accomplishments in FY'95

The project began by researching the literature to determine the current "state of the art" concerning navigational issues in a complex user interface. Several papers on the subject have been written by those involved in the Pad++ project from UNM, Bellcore, and NYU. Web Navigation was chosen as the application to investigate for the first year work. John Mareda, the PI, along with Heather Allen, a Sandia Human Factors expert investigated techniques to improve Web Navigation through data zooming. John installed several different versions of the Pad++ software on the SGI scientific visualization server machine at Sandia and on two Suns, one a SPARC 2 running OS 4.1.3 and the other a SPARC 20 running Solaris.

Glenn Machine assisted John in getting Pad++ put up as an application on TIE-In. Pad++ worked best when it runs on a local, fairly powerful machine. However, with a fast X-Server, Pad++ ran acceptably through TIE-In. John added a link to the EVE

home page as a pull down menu in Pad++ for evaluating data organization in HTML browsing. John and Heather met with Ben Bederson at UNM to discuss various approaches for organizing the HTML pages. Ben described the code that performs the HTML layout and John and Heather evaluated different layout approaches. John worked with Heather to access Pad++ through TIE-In on the Mac in her office running an X-Windows emulator. Because a fast X-windows display was critical to interactive response, using the Mac/X emulator is not viable. Heather used a UNIX workstation running X-windows for her portion of the work in evaluating HTML document arrangement.

A contract was placed with UNM for Pad++ support in areas of interest to Sandia. The first years contract was for $40K and was placed in May of 1995. A UNM graduate student rewrote the Pad++ based HTML browser in C++ making it more general, robust, and functional.

A three minute videocassette demonstrating several of the concepts of Pad++ was created using Sandia data. The video demonstrated semantic zooming in a directory tree and shows a preliminary HTML viewer written at UNM. TIE-In HTML pages were used in this demonstration. In this demonstration the user clicks on a link, and the subsequent page is brought up smaller and to the right of the original page. Through panning and zooming operations, one can navigate through the HTML data space. The history of the navigation session can be seen as the user zooms back out. The concepts of portals and filters are also demonstrated. A portal is a window which gives a view of some portion of the Pad++ data space. A filter is a special type of portal which can change the representation of the data which is viewed through the filter. For example, a chart of numbers could be viewed as a bar chart or a scatter diagram, using filters. In another example, changing the user input device from text entry, to a slider, to a dial gauge is demonstrated using filters.

John hosted a Sandia wide presentation/demonstration on Pad++ in the TTC, which was given by Jim Hollan and Ben Bederson of UNM. In addition to introducing this new interface technology to interested Sandians, a primary goal of the seminar was to solicit Sandia applications that could greatly benefit from the Pad++ technology for further research during the second year of this LDRD effort. Management of the SNL Website was identified as the best, opportunity for furthering this research in a direction that could beneficially impact SNL.

Accomplishments - in FY'96

Sandia has an exceptionally well designed and very complex Internal Website (Intranet). As Websites, like SNL's, grow in complexity and size, better tools are needed to design, develop and manage them. Because Pad++ is very well suited to navigating large data spaces made up of diverse data, it seemed natural to investigate using Pad++ as a tools for Website design and maintenance.

Dru Popper-Lopez, a Sandia staff member on the EVE team (Sandia's Internal Web), was added to the project to provide suggestions on what would be useful in a Website design and maintenance tool. He provided feedback on the utility of the tool as the prototype was developed.

We began the process by outlining what was involved in designing a Website. Some of the tasks involved in this process are to identify needs, define the target audience, write down the objectives of the site, outline the content, and create a rough map of the site. Although Website design and maintenance was chosen as the application to prototype, the same concepts apply to other storyboarding processes such as proposal writing, project planning, book writing, video production and interactive multimedia applications.

We identified desirable features for a Website Design tool based on multiscale user interface technology. The tool should have an intuitive combination of frame and GUI-based authoring. All design tasks should be interrelated, not separate elements. Templates should be provided that would make the tool easier to use. For example, when designing a new site from scratch, the user might be asked to fill in a three level block diagram. HTML, sound, video and image creation and editing tools should be available. The software should automatically start the appropriate application for editing a file, based on the MIME type. The tool should be useful for multiple levels of site design from creating a new site to retrofitting an old site, to generating a subset site from an existing site.

For existing sites, the tool should automatically create a layout of the site, which could be easily viewed using Pad++. For site maintenance, portal filters could be used to highlight specific file types. For example, an MPEG filter might turn all of the MPEG links yellow. Rearranging a site should be a simple matter of cutting and pasting with the re-linking being handled automatically by the software.

An analysis capability should be developed to determine if the site is meeting the original design goals. A weighting scheme should be developed to determine if the target audience is visiting the intended pages at the site. Pages and actions might be assigned values and the users visit could be scored. For example, if a user sends an email message, that visit could score a higher value. If a user sends in a check, that would score even higher. Based on the results, the site could be redesigned to meet the stated goals.

Other capabilities already supported in the Pad++ substrate that were used in this prototype are the directory browser and the HTML browser (including the camera view). Applying other research, such as the concepts in HyperG, also provided a more intuitive method for defining the structure of the site.

Varying amounts of text related to a link, can be displayed by using the zoom factor. From far away, the links in a document may appear as lines, as you get closer, additional information is provided, including keywords, titles and abstract information.

David Proft, was hired to lay out the structure of an existing site in Pad++. Ben noted that most sites are not laid out in a tree hierarchy and that having a filter show only certain types of links (forwards, backwards, internal only, external only, or all) would be of value, it was, also, suggested that keeping track of the path a user takes through the site would be useful. This path information would provide additional information measuring how easily a user is able to navigate through the site.

David Proft experimented with these concepts in Pad++, developed a prototype and documented the results in a progress report from UNM, which describes the zooming Website manager. Dru Popper-Lopez provided feedback and suggestions for changes that made the software more useful for Website design and maintenance.

Ed Marek researched other methods for representing the structure of a Website based on 3D techniques. Tamara Munzner and Paul Burchard researched visualizing the Web in 3D hyperspace. This idea grew out of research at the Geometry Center at the University of Minnesota. Images relating to their research (a conic tree and a 3D hyperbolic space) are shown below, Figure 1.

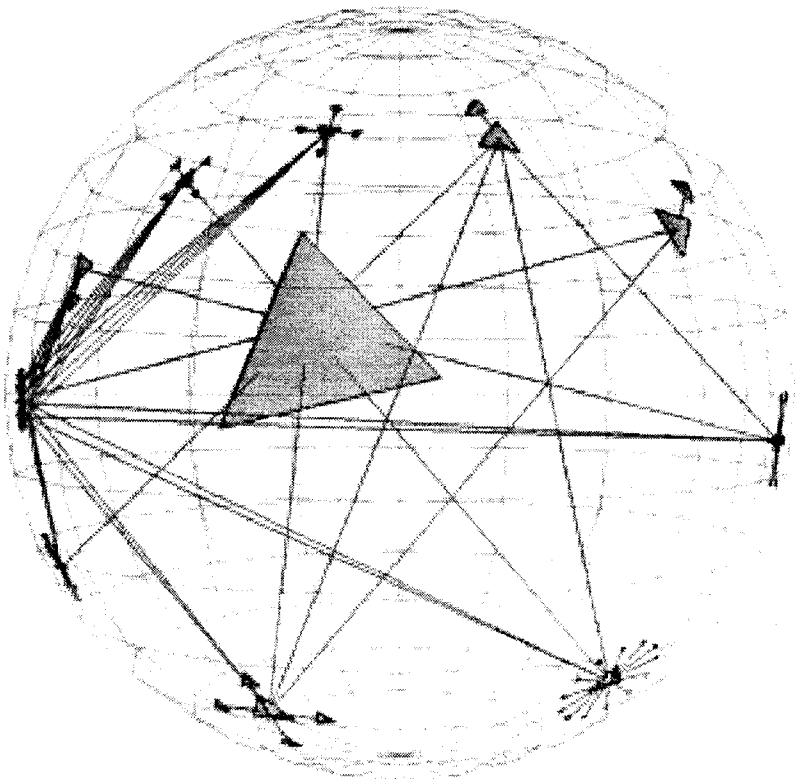


Figure 1, A visualization of a Website in 3D hyperspace

Ed also successfully investigated techniques for laying out a site in 3D using VRML. In the image below, a section of Sandia's external Web is represented by 3D objects in a 3D scene, see Figure 2.

An extensive series of human factors studies were undertaken using the Pad++ tool, testing the hypothesis that Pad++ improves a user's ability to retrieve, comprehend, and integrate data from the WWW. The results indicated that subjects can perform browsing tasks 23% faster, on average, using Pad++ than when using Netscape. Additionally, many subjects found Pad++ intuitive and fun to use. The experimental details and results are presented in Appendix I, which reproduces pertinent sections of the human factors study report in "A Zooming Web Browser".

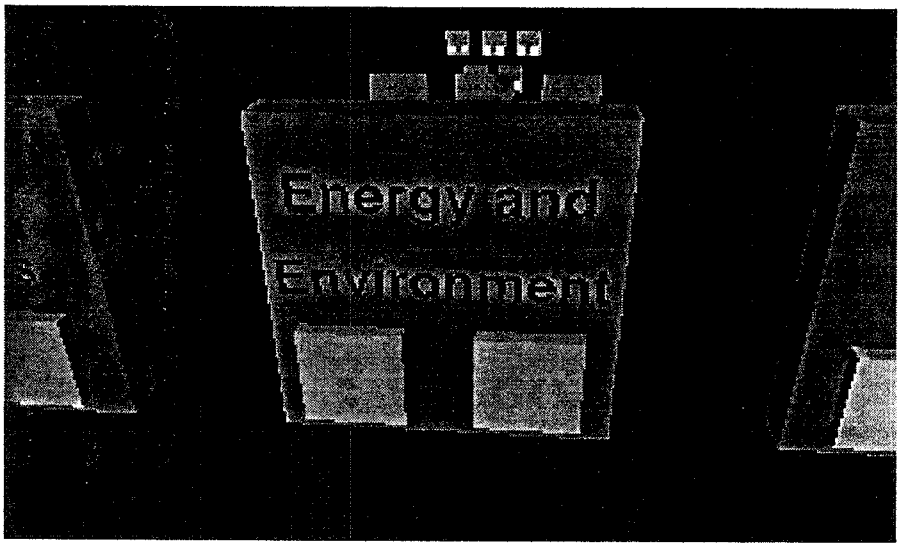


Figure 2, A visualization of a Website in 3D space using VRML

Conclusion

Three dimensional user interfaces are very useful tools for complicated data sets such as Websites. Beyond three dimensional viewers (for example, VRML) the infinite resolution Pad++ viewer makes it even easier to create and maintain large websites. The use of filters and portals were especially useful. In conclusion, Pad++, and this research in general, point to a new class of human/computer interfaces. With the advent of inexpensive high performance personal computers, these more computationally demanding interfaces, or ones derived from them, will become standard tools.

Appendix I

This appendix discusses the human factors studies. This work was reported by Benjamin B. Bederson, James D. Hollan Jason Stewart, David Rogers, David Vick from UNM and Laura Ring, Eric Grose, Chris Forsythe from Sandia National Laboratories. References from "A Zooming Web Browser" are included on the last page of Appendix I.

USABILITY TESTING

We hypothesize that the Pad++ Web browser will improve users' abilities to retrieve, comprehend, and integrate data from the WWW. We think that the ability to see multiple pages simultaneously, and to learn the structure of the Web will facilitate tasks of this nature. We chose to study the Pad++ Web browser in a focused technical domain to make it easier to evaluate, by allowing measurement of times to answer simple questions. In addition, this domain is representative of the use of a Web browser within a product development environment.

The Stimuli

The usability testing was designed to test how well the Pad++ Web browser could perform as a viewer for this type of database. More specifically, we created a visual database of Web pages. We then compared how long it took to access information from these pages using the two browsers, Pad++ and Netscape.

We designed a hypothetical database containing the evolution of a manufacturing process over a series of six versions. The entire stimuli set consists of 31 Web pages (Figure 4). The stimuli contains an *Overall* page which contains information on the database and links to the *Design and Manufacturing Process* pages and links to the *Budget and Schedule* pages for each of the six versions. Six *Budget and Schedule* pages (Figure 5) give an overall estimate of time and the budget for each version. Six *Design and Manufacturing Process* pages contain the process flowcharts for each version (Figure 6). The process flowcharts are WWW image maps which link to notes about the process. The final 18 pages contain these notes.

Task 1 required the subjects to navigate to specified pages in the data base and enter a page number found on the top of each page. For example, the subjects were asked to enter the page number of Version 5's *Budget and Schedule* page. This tested speed of navigation within the browser.

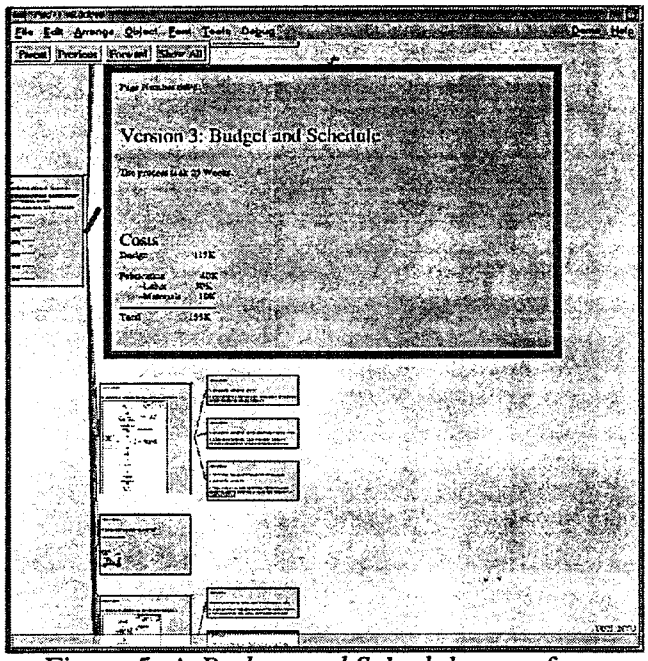


Figure 5: A *Budget and Schedule* page from stimuli

Task 2 and Task 4 tested the users ability to retrieve and comprehend information about the whole database. In Task 2 the subjects were asked to count the number of versions with certain features or where certain events occurred. For example, the subjects were asked to indicate the number of versions that had a new step added. Task 4 required the user to identify versions with certain features or where certain events occurred. For example, the subjects were asked which version had a problem with fabrication.

Task 3 tested the users ability to retrieve and integrate information found within different pages of the database. The subjects were asked to compare different versions. For example, the subjects were asked to indicate which version, out of versions one, two and three had the highest material cost. This required the subject to look at the budget and schedule page for the three versions and compare each of their material costs.

The Experimental Conditions

Four experimental conditions were examined: *Pad++ Prebuilt, Pad++ Interactive, Netscape Regular* and *Netscape Graphic*. In the *Pad++ Prebuilt* condition, the entire network of pages was loaded into Pad++ prior to beginning the task. In the *Pad++ Interactive* condition, only the *Overall* page was loaded and the network of pages was built as the user answered the questions. In the *Netscape Graphic Condition* a WWW image map of the entire database was created by doing a screen capture in Pad++. This condition was added to see if Pad++'s navigational tools caused improvement in performance independent of that caused by the visual representation of the database. The image map used contains a snapshot of the pages depicted in Figure 4.

Training

If a subject reported using Netscape less than 2 hours a week, they were asked to complete a brief training task. The task consisted of browsing through a set series of pages from the University of New Mexico's Web Site. This took approximately 10 minutes. During this task the subjects were instructed on the use of the "forward" and "back" buttons, and how to click on hypertext and image maps in order to follow links.

All but one subject had never used Pad++, although a few had seen brief demonstrations of the software. Three training tasks were used to familiarize the subjects with panning and zooming in Pad++ and using the Pad++ browser. Training tasks 1 and 2 were used to familiarize the subjects with a three-button mouse. In training task 1 the subjects were asked to zoom in and out. In training task 2 the subjects were asked to zoom, pan and select objects. The subjects performed both tasks until they were able to perform 10 operations without error. The third training task asked the subject to open a series of pages within the University of New Mexico's Web Site and then navigate to different open pages. During the training, the subjects were informed of different strategies to use when navigating in the Pad environment.

Subjects

The experiment was carried out at the University of New Mexico. There were 32 participants, but only 30 participants yielded usable data. The two participants whose data was not used had over 50% of the questions incorrect. All of the subjects had completed at least two years of college and all use a computer at least two hours a week. The majority of subjects, 73%, used a computer more than an hour a day. 93 percent reorganized of the subjects had browsed the World Wide Web before, and 80% of subjects browsed the World Wide Web at least an hour a week. 90 percent of the subjects were students and the other 10% were professionals.

Procedures

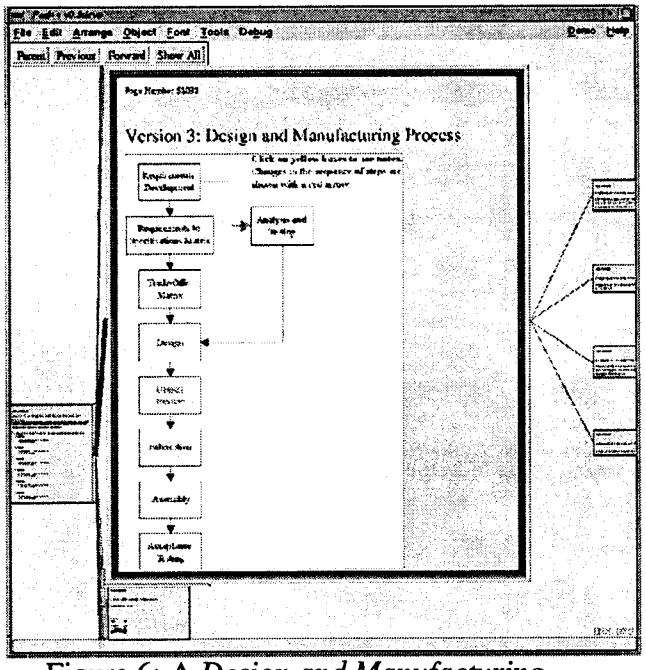


Figure 6: A *Design and Manufacturing* Process page

Prior to beginning the experimental tasks, the subjects were trained if needed and then asked to read through an instruction page. The instruction page explained the task and database, and had links to example pages. The subject was instructed to click on all links and look at all the examples. The subjects then carried out trials for the four experimental tasks. In one window, the browser was open with the appropriate initial pages, and in another window, the questions were displayed. A program written in Tcl displayed the questions, and recorded the time spent on each question and the subjects' response. After each question, the entire screen went blank so they could not see the browser and database pages. The subjects were instructed to click on a bar to proceed. This action started the timer and displayed the question and the browser window. Subjects clicked on another bar to enter their response and stop the timer. After the experimental task, a questionnaire was displayed which asked questions about the browser, for information on computer usage and for some demographic information.

Results

In order to determine the quality of the questions, we looked at the percentage of subjects who answered each question correctly. If over 20 percent of the subjects missed the question, it was considered too difficult and thrown out of further analysis. Three questions out of 24 were thrown out for this reason. Two of these questions were in Task 2 and one of these questions was in Task 3.

The dependent variable was the average time it took the subjects to answer each question. Only the times for questions answered correctly were used in calculating the average time. An average time to correct response was calculated first for each of the four tasks. The average times collapsing across tasks were also calculated. The number of correct responses ranged from 16 out of 21 to 21 out of 21 with the majority of subjects having one or fewer incorrect responses.

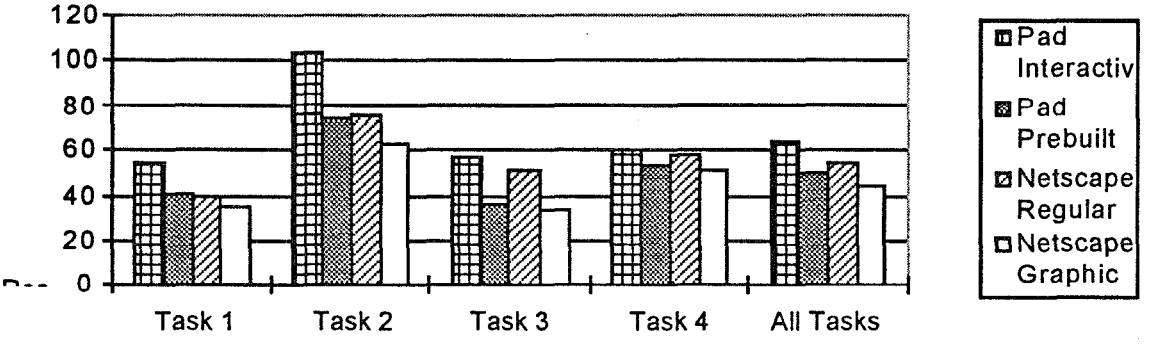
A one-way analysis of variance was performed on each task's average time to correct response. We are aware that in using an alpha of 0.10 for each test, we may be inadvertently increasing our Type I error rate, but it is necessary to do so in order to have enough statistical power given the limited number of subjects.

Figure 7 shows a graphical representation of the average time to correct response for each task and collapsed across tasks. The overall F test was only significant for Tasks 1 and 2, using an alpha level of 0.10, $F_{3,26}=2.74$, $p = 0.064$ and $F_{3,25}=2.66$, $p=0.070$ respectively. These results were significant because subjects in the *Pad++ Interactive Condition* took longer to carry out the tasks than did the subjects in the *Netscape Graphic* condition.

These results were largely attributable to technical problems with Pad++ that made it run considerably slower than Netscape. Based on the qualitative data collected by observation and from the subjects responses to the questionnaire, we made various improvements to the Pad++ browser described in the next section.

The speed at which Pad++ was running caused some discontent, but many comments suggested

Figure 7: Average Time to Correct Response by Task and Browser Condition after the First Experiment



that subjects felt the interface to be intuitive and fun to use. Some of the quotes that indicate the positive aspects of Pad++ follow:

ZOOM is fun!

The fact that one can get the "whole picture" in varying degrees of closeness is fantastic.

I love the zoom in and out. I also liked Show All very much.

The layout of the pages made the task very easy once one figured out that the pages were laid out in the order listed on the parent page. With the color scheme used, the zooming browser allowed answering of some questions merely by panning and examining the pages from a low resolution view.

Easy to find what I need to know. I did not have to search very much.

I liked the tree structure, but disliked the occasional delays in updating the display. Once I knew where the pages were in the tree structure, I could quickly find what I wanted.

Another interesting observation made by the experimenter during testing was that subjects varied in their use of panning and zooming available in Pad++. Due to their familiarity with browsing in Netscape, they seemed reluctant to try some of the other features available to them in Pad++. They stuck to what they knew, despite that in training, the experimenter encouraged the subjects to use all available navigational tools. Furthermore, it appeared that there was a positive relationship between subjects use of these navigation features and task performance. In future studies it would be interesting to quantify this difference between subjects and to see if it is positively correlated with task performance.

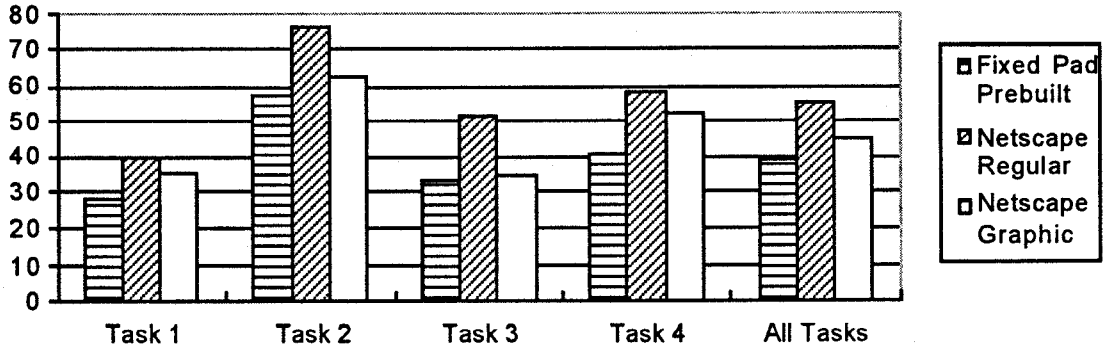
The Improvements Made to the Pad Browser

1) We made Pad++ run roughly twice as fast by using an 8-bit instead of a 16-bit X-server.
2) When a page was brought into focus it initially was aligned with the left side of the screen. Some subjects pointed out that this left them feeling disoriented due to a loss of context. We changed this feature so that the page would be centered when it was in focus.
3) The fonts were improved from being mediocre to using high-quality system fonts.
4) During testing, subjects pointed out that they wanted to be able to defocus a page as well as focus it. A feature was added so the user need only click again on the page to bring it out of focus and make it small again.
5) We made pages thinner so more context could be seen. This was possible because of the improved font quality.
6) We fixed a cursor bug which only occurred when we tested the software using novice users. This bug caused the cursor to freeze on a zooming cursor if the user pressed two mouse buttons at once.
7) Some subjects indicated that they missed the Netscape feature of the cursor changing when it was over a link. This feature was added in Pad++, so the hand cursor changed into an arrow.

Experiment 2

Following the first experiment, we realized that we needed to make some changes in order to evaluate Pad++ more fairly. First of all, we made the

Figure 8: Average Time to Correct Response for Improved Pad++ versus the Netscape Conditions by Task



above mentioned improvements. Secondly, since almost all of the subjects in the Netscape conditions were regular Netscape users, we felt that to evaluate Pad++ in a comparable manner, we should use subjects that were familiar with Pad++.

We ran the *Pad++ Prebuilt* condition again with seven Pad++ users using the same procedures as discussed previously. All seven of these subjects worked as part of the Pad++ development team. All were familiar with navigation in the Pad++ environment, and did not need to be trained.

Results

A one-way analysis of variance was performed to compare the new subjects performance to those subjects in the *Netscape* conditions. The average time to correct responses for each task and averaged across all the tasks are displayed in Figure 8.

The overall F-test approached significance for Task 1 and for the average across all tasks, $F_{2,19}=2.33$, $p=0.124$ and $F_{2,19}=2.43$, $p=0.115$ respectively. Furthermore, tests of the a-priori hypothesis that Pad++ would do better on average than the *Netscape* conditions yielded a p-value of 0.0599, $F_{1,19}=4.00$ for Task 1 and a p-value of 0.088, $F_{1,19}=3.23$ for the average time to correct response across all tasks.

These results demonstrate that subjects can perform this browsing task 23% faster on average using the Pad++ browser than using Netscape. Many subjects also found the Pad++ browser intuitive and fun to use. In a manufacturing setting, use of the Pad++ browser could increase workers productivity by speeding up the time it takes them to access historical information about a product's process development over time. Furthermore, since using the Pad++ browser allows the user to access such information in a fun and intuitive manner, it may lead to an increased use of historical information with consequent improvements in product and process quality.

CONCLUSION

The World-Wide Web has become an important and widely used resource. Because of this, it is crucially important to address its usability. We have shown one promising technique based on zooming to better support Web navigation. This technique was used to implement a prototype zooming Web browser that was found to be faster at accomplishing specific tasks than a traditional browser.

Pad++ and the zooming Web browser will be made generally available in the near future. To find current information, send mail to `pad-info@cs.unm.edu`, or look at `<http://www.cs.unm.edu/pad++>`.

ACKNOWLEDGEMENTS

We acknowledge generous support from DARPA's Human-Computer Interaction Initiative (Contract #N66001-94-C-6039). This work also supported by the United States Department of Energy under Contract DE-AC04-96AL85000. We appreciate the work of our collaborators at the New York University Media Research Lab, especially that of Jon Meyer, in helping us develop Pad++.

REFERENCES

1. Benjamin B. Bederson and James D. Hollan. "Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics", *Proceedings of UIST'94* ACM Press, 17-26.
2. Benjamin B. Bederson, James D. Hollan, Ken Perlin, Jon Meyer, David Bacon, and George Furnas. "Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics", *Journal of Visual Languages and Computing* (7), 1996, 3-31.
3. Stuart K. Card, George G. Robertson, and William York, "The WebBook and the Web Forager: An Information Workspace for the World-Wide Web", *Proceedings of CHI'96*, ACM Press, 111-117.
4. Bay-Wei Chang and David Ungar. "Animation: From Cartoons to the User Interface", *Proceedings of UIST'93*, ACM Press, 45-55.
5. Peter Doemel, "WebMap - A Graphical Hypertext Navigation Tool", *2nd International Conference on the World-Wide Web*, Chicago, IL, 1994, 785-789.
6. William C. Donelson. "Spatial Management of Information", *Proceedings of SIGGRAPH'78*, ACM Press, 203-209.
7. George W. Furnas. "Generalized Fisheye Views", *Proceedings of CHI'86*, ACM Press, 16-23.
8. George W. Furnas and Jeff Zacks. "Multitrees: Enriching and Reusing Hierarchical Structure", *Proceedings of SIGCHI'94*, ACM Press, 330-336.
9. Wendy A. Kellogg and John T. Richards. "The Human Factors of Information on the Internet", *in Advances in Human Computer Interaction*, Ed. J. Nielsen, Ablex Press, (5), 1-36.
10. John Lamping, Ramana Rao, and Peter Pirolli. "A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies", *Proceedings of CHI'95*, ACM press, 401-408.

11. Jock D. Mackinlay, George G. Robertson, and Stu K. Card. "The Perspective Wall: Detail and Context Smoothly Integrated", *Proceedings of CHI'91*, ACM press, 173-179.

12. Sougata Mukherjea, James D. Foley, and Scott Hudson. "Visualizing Complex Hypermedia Networks through Multiple Hierarchical Views", *Proceedings of CHI'95*, ACM press, 331-337.

13. John K. Ousterhout. *Tcl and the Tk Toolkit*, Addison-Wesley, 1994.

14. Ken Perlin and David Fox. "Pad: An Alternative Approach to the Computer Interface", *Proceedings of SIGGRAPH'93*, ACM Press, 57-64.

15. Monojit Sarkar and Marc H. Brown. "Graphical Fisheye Views", *Communications of the ACM*, 37 (12), December, 1994.

16. Maureen C. Stone, Ken Fishkin, and Eric A. Bier. "The Movable Filter as a User Interface Tool", *Proceedings of SIGCHI'94*, ACM Press.

17. David Ungar and Randy B. Smith. "Self: The Power of Simplicity", *Proceedings of OOPSLA'87*, 227-241.

18. Kent Wittenburg, Duco Das, Will Hill, and Larry Stead. "Group Asynchronous Browsing on the World Wide Web", *Proceedings of the 4th International World Wide Web Conference*, Boston, MA, International WWW Conference.

Appendix II

Data Zooming Project Members

Project members at Sandia

John Mareda - PI

Responsibilities - Project leader. Determine goals of the project and direct the project members towards achieving those goals. Conduct project meetings, report status and progress.

Level of effort - 0.4 FTE

Deliverable - Project (SAND) report in HTML format by April 30, 1996

Contact information - jfmared@sandia.gov, 845-8550

Ed Marek - PM

Responsibilities - Program Manager. Manage the project funds. In addition, make technical contributions in the area of application design. Also, compare and contrast Pad++ software with other leading edge visualization technologies, such as VRML.

Level of effort - 0.1 FTE

Deliverable - Input to project report by April 15th

Contact information - elmarek@sandia.gov, 845-8550

Dru Popper-Lopez

Responsibilities - Dru is a member of Sandia's EVE team (Sandia's internal Web site). She is responsible for providing input to the design of the Web site design and maintenance application. She will continue to work with UNM providing feedback as they implement the prototype application in Pad++.

Level of effort - 0.1 FTE

Deliverables - Input to project report by April 15th. Continued interaction with Ben Bederson through September 30th. Input to Ben Bederson for final report by September 15th.

Contact information - dpopper@sandia.gov, 822-2239

Heather Allen

Responsibilities - Heather is an expert in human factors issues for user interfaces at Sandia. She is responsible for evaluating the design and providing feedback on the prototype Website design and maintenance application from a human factors perspective. She will continue to work with UNM providing feedback as they implement the prototype application in Pad++.

Level of effort - 0.1 FTE

Deliverables - Input to project report by April 15th. Continued interaction with Ben Bederson through September 30th. Input to Ben Bederson for final report by September 15th.

Contact information - hwallen@sandia.gov, 844-6657

Project members at UNM

Ben Bederson

Responsibilities - Ben is an assistant professor at UNM and is in charge of the Pad++ project. He oversees the application development at UNM, including coordinating the work of the UNM student involved in this project. He is also responsible for continuing research into this new user interface technology.

Level of effort - Approximately 25% through September, 1996

Deliverables - Input to project report by April 15th. Continued lead of UNM programming effort through September 30th. Final project report in HTML format by September 30, 1996

Contact information - bederson@cs.unm.edu, 277-3914

David Proft

Responsibilities - David is a graduate student at UNM. He is responsible for implementing the storyboarding application in the Pad++ substrate

Level of effort - Approximately 50% during the spring and 100% during the summer semester

Deliverables - Input to project report by April 15th. Continued interaction with Ben Bederson through September 30th. Input to Ben Bederson for final report by September 15th.

Contact information - proft@cs.unm.edu, 277-1481

Appendix III

Installing Pad++

The following steps document the installation procedure for installing Pad++ on a virgin Sun SPARCstation 20 (right out of the box). These instructions assume you are relatively familiar with working in and installing software in a UNIX environment. The versions of Pad++ and the required software such as TK/TCL change often. Refer to the Pad++ installation instructions to determine which versions of the software are required.

C Compiler

1. Order and install C compiler from Sun. It came on a SunSoft WorkShop CD. The installation instructions are on the CD-ROM jacket. You'll need to get a license from Sun. Just for good measure, I also installed C++, but I do not believe that is necessary to install Pad++.

GNU Software

To retrieve the various GNU files type "ftp gatekeeper.dec.com", log in as anonymous and use your user id as your password. Type `cd pub/GNU', to go into the GNU directory and type `binary' to cause ftp to perform binary file transfers.

2. Install the Gnu gzip utilities. A lot of gnu software will need to be installed and the files are stored in gzip compressed form, so the gzip decompression utilities are the first thing you will need to install. Get the first file with the command `get gzip-1.2.4.tar'. Extract the files with `tar xvf gzip-1.2.4.tar'. Go into the gzip-1.2.4 directory and type `./configure', `make', and as Superuser `make install'. This will create a /usr/local/bin directory and put the gzip compression files in that directory.

This site also has pre-compiled binaries for these utilities. They are located in ftp://gatekeeper.dec.com/pub/GNU/sparc-sun-solaris2. Since I did not discover this until after I had already built the gzip utilities, I just used the ones I had built.

3. Next I retrieved patch-2.1.tar.gz, typed `gunzip patch-2.1.tar.gz' to untar the file, went into the patch-2.1 directory and typed `./configure', `make', and as Superuser `make install'.

4. Repeat step 3 for bison-1.24.

5. Repeat step 3 for make-3.74.

6. I followed the above procedure for gcc-2.7.0, but it failed. Here you'll need to follow the instructions in the INSTALL file. When I did it originally, I got an error `fatal: file cccp: cannot msync file; errno=28'. I tried doing the make using both the bootstrap approach and doing it separately and got the same error both times. Using a shotgun trouble shooting approach, I changed to root with SU, copied the tar file (gcc-2.7.0.tar) to /usr/local and untared the file as Superuser. I then made sure ucb was not in my path. I then did the build separately, using the command `make CC="TERMCAP=x OBJS=x LIBFUNCS=x STAGESTUFF=x cc" LANGUAGES=c'.

Some of these hints were in the Installing GNU CC on the Sun section of the INSTALL file. After that I followed the steps in the rest of the INSTALL file. I typed `make stage1', `make CC="stage1/xgcc -Bstage1/" CFLAGS="-g -O" LANGUAGES=c' (no fatal errors, yea). Having met with such tremendous success, I decided to follow the procedure to test the compiler. I typed `make stage2' followed by `make CC="stage2/xgcc -Bstage2/" CFLAGS="-g -O" LANGUAGES=c'. I then typed `make compare' No differences. Great. I then typed `make install CC="stage2/xgcc -Bstage2/" CFLAGS="-g -O" LANGUAGES=C' to install the GNU C compiler.

I repeated the above process to build and install c++. It wasn't until I tried to install pad++ that I discovered I needed g++. I had bought Suns c++ compiler so I guess it was wishful thinking on my part that pad++ would use that compiler. I suspect if I would have said LANGUAGES=c, c++ when I did the original build all would have been fine.

TK/TCL

7. Type `ftp ftp.cs.berkeley.edu' and log in as anonymous and use your user name as your password. Type `binary', `cd ucb/tcl', and `get tcl7.3.tar.Z'. Exit ftp, type `gunzip tcl7.3.tar.Z' and `tar xvf tcl7.3.tar'. Go into the tcl7.3 directory and type `./configure', `make', and as superuser `make install'.

8. Type `ftp ftp.cs.berkeley.edu' and log in as anonymous and use your user name as your password. Type `binary', `cd ucb/tcl', and `get tk3.6.tar.Z' and `get tk3.6p1.patch'. Exit ftp, type `gunzip tk3.6.tar.Z' and `tar xvf tk3.6.tar'. Move the file tk3.6p1.patch into the tk3.6 directory. Go into the tk3.6 directory and type `patch < tk3.6p1.patch'. This will install the patch. Then type `./configure', `make', and as superuser `make install'.

Pad++

9. That's right, now I remember, I wanted to install Pad++. To perform this step you will have had to gotten permission to retrieve the pad-0.2.2.tar.gz file. Once you have this file, follow the same procedure as the preceding steps for installing software. Type `gunzip pad-0.2.2.tar.gz' followed by tar xvf pad-0.2.2.tar. Follow the instructions in the Pad++ installation directory. If all is well, you should be able to type "./configure", "make", and as Superuser "make install" to install pad on the target machine.

DISTRIBUTION:

5 University of New Mexico
Attn: Benjamin B. Bederson
Attn: James D. Hollan
Attn: Jason Stewart
Attn: David Rogers
Attn: David Vick
Computer Science Department
Farris Engineering Center Room 313A
Albuquerque, NM 87131

1	MS-0188	C. E. Meyers, 4523
1	MS-0318	G. S. Davidson, 9215
1	MS-0318	J. F. Mareda, 9215
1	MS-0318	V. Holmes, 9215
1	MS-0829	L. T. Ring, 12323
1	MS-0829	E. M. Grose, 12323
1	MS-0829	J. C. Forsythe, 12323
1	MS-1110	C. Pavlakos, 9215
1	MS-9018	Central Technical Files, 8523-2
5	MS-0899	Technical Library, 4414
2	MS-0619	Review & Approval Desk, 12630 For DOE/OSTI